Czech technical university in Prague

Course: BI-VWM

Semester: Summer 2021/22

Hypertext search

PageRank based content ranking

Šimon Melkes (melkesim) | Peter Kosorín (kosorpet)

Contents

List of Figures

1 Project description

The aim of this project was to implement a simple search engine which demonstrates the algorithms and calculations used in ranking web pages based not only on their full-text similarity to a query but also based on their importance in the web graph. The project consists of a web crawler which is used to download each page, a module which calculates the PageRank of each saved site, a module which calculates the text similarity at query time and a web interface. The domain chosen for crawling was Wikipedia as each article contains relatively large amount of outlinks.

2 Solution methods

2.1 Web Crawler

The web crawler module was implemented as a multi-threaded runnable which receives a starting page, then each thread independently browses outlinks leading to other pages, saving their HTML representation in the process, until a set number of sites or a maximum crawling depth is reached. In our case the maximum number of sites was set to 1000 with depth of 3. Each site is then assigned a unique ID. Lastly, each site is stripped of menus which are present on every Wikipedia article, as these outlink would skew the PageRank calculation.

2.2 Page rank calculation

Once a sufficient number of websites has been saved, the next step is creating an adjacency list for each page, each site remembers its own ”neighbor list”. Then a stochastic adjacency matrix **S** is created, where

$$S_{ij} = \frac{1}{|P_i|},\ if\ P_i\ has\ a\ link\ to\ P_j,\ else\ 0$$

(where $|P_i|$ is the number of P_i's outlinks). If a site has no outlinks (for example a picture or just plain text), the whole row is filled with value $\frac{1}{n}$ where n is the number of all web pages. Next, the matrix **G** is calculated, where

$$G = \alpha S + (1-\alpha)\frac{1}{n}ee^T,\ where\ \alpha \in (0,1)\ is\ a\ parameter\ (damping\ factor).$$

The matrix **G** is now positive and completely dense.
The final PageRank vector is obtained iteratively using the following formula:

$$\pi^{(k+1)T} = \pi^{(k)T} G.$$

The iterative process is stopped once the values have converged.

2.3 Text similarity calculation

The text similarity module is handled by the Apache Lucene library, which indexes all of the downloaded web sites. A query similarity score is assigned to each site at the time of the query. The library calculates the score depending on multitude of factors, including the term frequency in the document and number of terms in the query that were found in the document.

3 Implementation

The project was implemented in the Java programming language. For web crawling and HTML parsing the JSoup library was used. For matrix multiplication and matrix related operations we used the Efficient Java Matrix Library (EJML). For text similarity and scoring we used Apache Lucene. Finally, the web interface is handled by the Spring Boot package.

4 Usage example

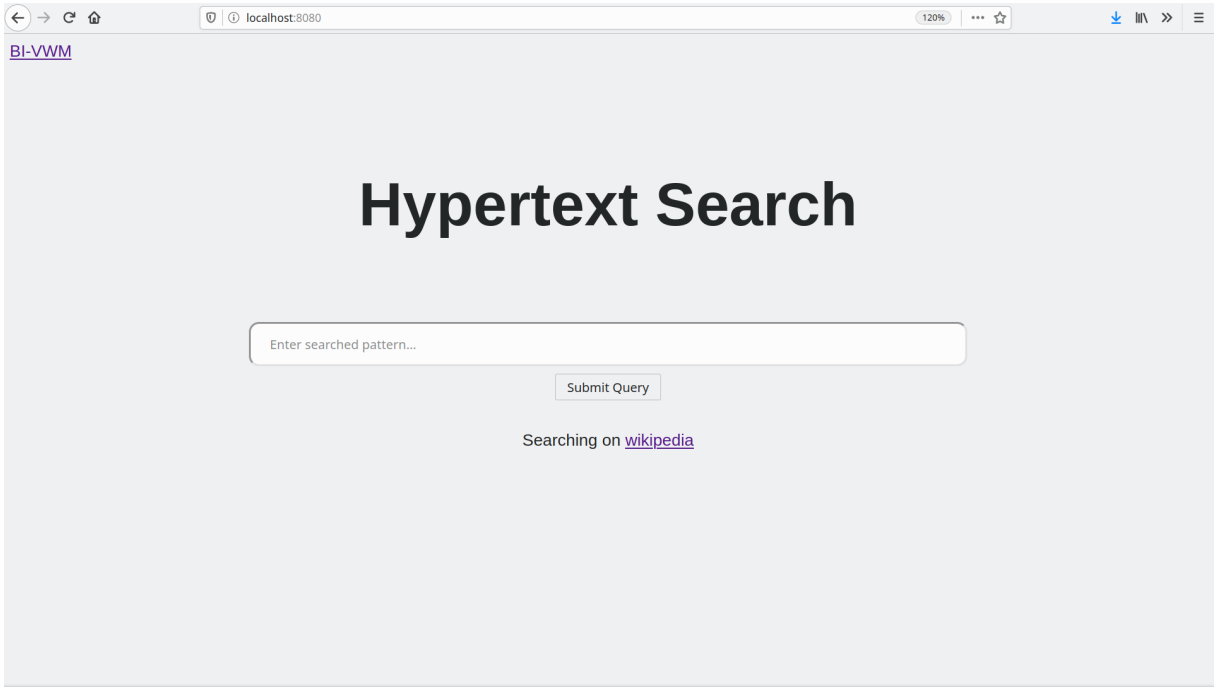

Figure 1: Main search screen

As shown in Figure 1, the main search screen contains a search bar where a text query can be entered.

The result screen (Figure 2) contains two columns of results, one on the right sorted solely according to text similarity, the one on the left sorted using both the PageRank and text similarity

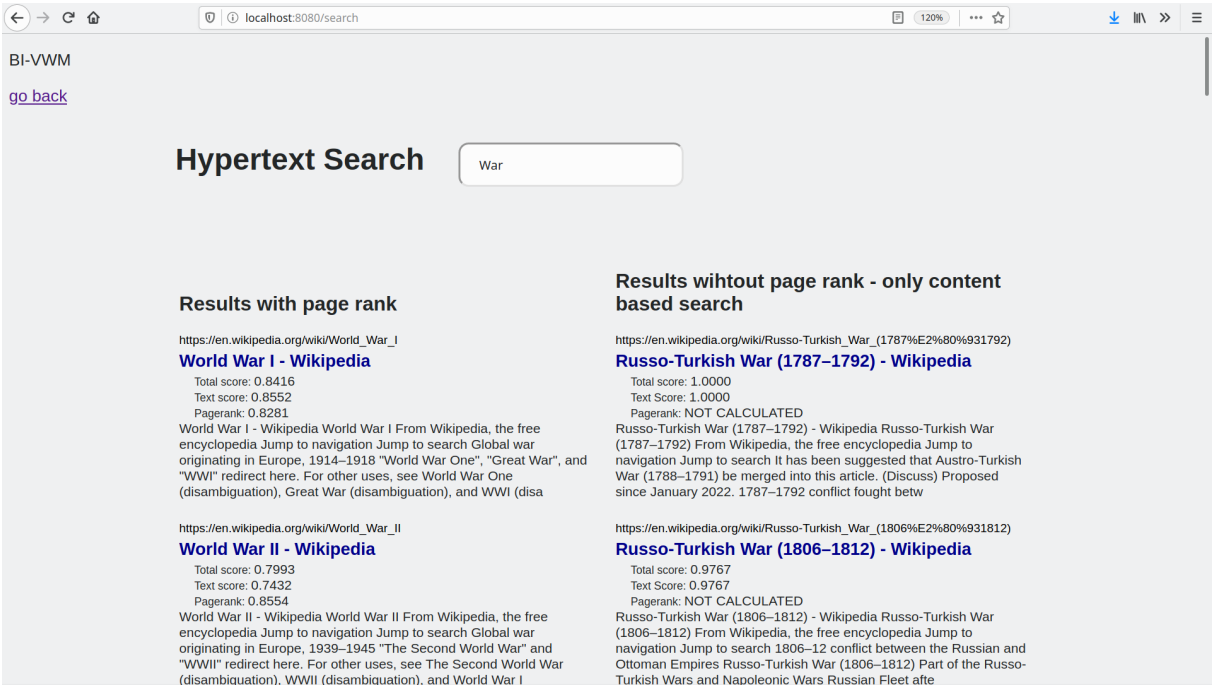

Figure 2: Result screen

score. This shows that the World War I article is more important than the articles about Russo-Turkish Wars.

5 Experiments

5.1 Page rank vector convergence

The iterative nature of the page rank calculation offers the question of how many iterations are needed for the values to converge. The following graph shows the change in the number of iterations depending on the decimal precision.

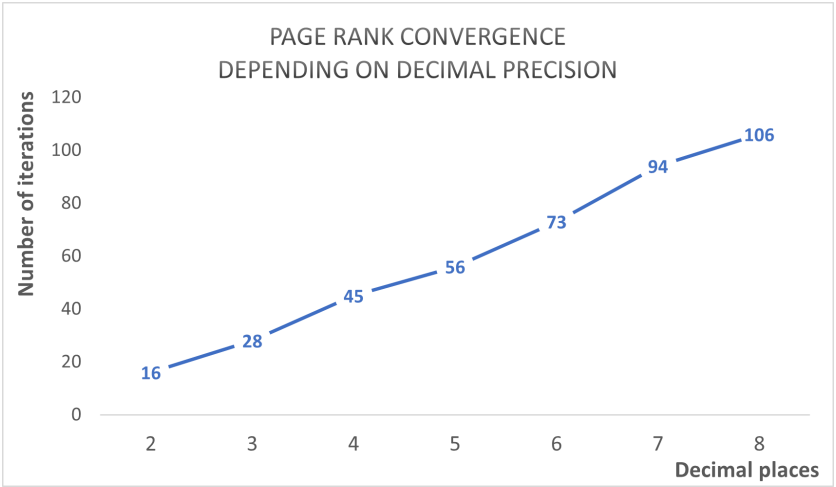

Figure 3: Iteration graph

5.2 Link matrix calculation

The next experiment shows the time needed for the calculations described in section 2.2, particularly the creation of the link matrix **G**. The following table shows the time needed.

Matrix size [squared]	100	250	500	1000	1500	2000
Time [ms]	2	9	10	22	35	66

Figure 4: Time table

The relatively fast calculation times can be attributed to to EJML library which, as the name suggests, is very efficient.

5.3 Damping factor

The damping factor α, which is the click-through probability, is included to prevent sinks (i.e. pages with no outgoing links) from ”absorbing” the PageRanks of those pages connected to the sinks. An infinite surfer would have to end up in a sink given enough time, so the damping factor allows a heuristic to offset the importance of those sinks.
When the damping factor is set close to 1 (for example 0.99) the pages with no outlinks gather the largest PageRanks. As the damping factor approaches 0 (0.01) then all clicks are basically random restarts, which means the page rank gets uniformly distributed among all the sites.

6 Discussion

The current bottleneck of the project is the limited heap space to which the websites were saved. Experiments showed that the upper limit of sites that can be saved is approximately 3000 before a heap space exception is thrown. The current implementation is also not scalable. The direct matrix computations used in this project would not be feasible at scale.

7 Conclusion

The project has successfully shown the advantage of sorting web content query results based on text similarity accompanied by a PageRank score. However in a real world scale application, other methods need to be used.